9

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/464,636	12/15/1999	RICHARD DIEVENDORFF	3382-49606	7885

7590 07/07/2003

KLARQUIST SPARKMAN CAMPBELL
LEIGH & WHINSTON LLP
ONE WORLD TRADE CENTER SUITE1600
121 S W SALMON STREET
PORTLAND, OR 97204

EXAMINER	
ZHEN, LI B	
ART UNIT	**PAPER NUMBER**
2126	14

DATE MAILED: 07/07/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

9

Office Action Summary	Application No.	Applicant(s)
	09/464,636	DIEVENDORFF ET AL.
	Examiner	Art Unit
	Li B. Zhen	2126

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____ .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-16 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) 13 is/are allowed.

6)☒ Claim(s) 1-12 and 14-16 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 7-13 .	6) ☐ Other: .

DETAILED ACTION

Allowable Subject Matter

1. Claim 13 is allowed.

Information Disclosure Statement

2. The information disclosure statement filed July 16, 2001 (paper no. 8) fails to comply with 37 CFR 1.98(a)(2), which requires a legible copy of each U.S. and foreign patent; each publication or that portion which caused it to be listed; and all other information or that portion which caused it to be listed. It has been placed in the application file, but the information referred to therein has not been considered.

Specification

3. Applicant referred to a plurality of references in the specification: p. 3, lines 21 – 44; p. 28, line 23 – 26; p. 30, lines 17 – 19. These references are not checked. The examiner requests a copy of the references so that they can be fully considered.

Double Patenting

4. Claims 1 – 5 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 5 of U.S. Patent No. 6,425,017 (herein after Dievendorff) in view of U.S. Patent No. 6,253,256 to Wollrath.

As to claim 1, Dievendorff teaches an object configuration store containing object properties information representing properties of at least first and second object classes executable in the system, the object properties information designating the first and second objects as supporting queued method invocation, the second object class having a method with a parameter for passing an object reference (claim 5, lines 15 – 19);

a method invocation recording facility operative responsive to request of a client program to supply method invocations recorders for object instances of object classes designated as supporting queued method invocations (claim 5, lines 20 – 24);

a first method invocations recorder supplied by the method invocation recording facility at request of the client program for the first object class (claim 5, lines 25 – 30);

a second method invocations recorder supplied by the method invocation recording facility at request of the client program for the second object class, the second method invocations recorder operating in response to a method invocation in which an object reference to the first method invocations recorder is passed to cause a data stream representation of the first method invocations recorder to be marshaled into a method invocations message for submission into a message queue associated with the second object class (claim 5, lines 25 – 34);

a method invocation play-back facility operative responsive to a queued method invocations message to supply method invocation players for object instances of object classes designated as supporting queued method invocations (claim 5, lines 35 – 39); and

a first method invocation player supplied by the method invocation play-back facility in response to the method invocations message queued to the message queue associated with the second object class, the first method invocations player operating in response to the method invocations message to unmarshal the date stream representation (claim 5, lines 40 – 48). As to a second invocation recorder, obviously multiple method invocation recorders could be created.

As to a object class having a method with a parameter for passing an object reference, including a data stream representation of the method invocations recorder to be marshaled into a method invocations message, and creating a copy of the first method invocations recorder, these limitations are taught by Wollrath, see the rejection to claim 1 below.

It would have been obvious to a person of ordinary skilled in the art at the time of the invention to apply the teaching of Wollrath to the invention of Dievendorff because the invention of Wollrath provides efficient transfer of objects in a distributed system such that when an object is needed at a remote machine, the object can be transmitted and reconstructed at the remote machine (column 8, lines 51 – 65 of Wollrath).

5. Claims 14 – 15 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 11 of U.S. Patent No. 6,425,017 (herein after Dievendorff) in view of Wollrath.

As to claim 14, Dievendorff teaches a component recorder constructor operating on request of a client program to obtain a component reference to create a method invocation recording component and return a reference for such method invocation recording component to the client program (claim 11, lines 42 – 47);

a first method invocation recording component created by the component recorder constructor (claim 11, lines 48 – 50); and a second method invocation recording component created by the queued component recorder constructor responsive to a second request of the client program to obtain a second reference for a second queued component, the second method invocation recording component operating in response to an invocation of the reference passing method made on the second method invocation recording component having a reference for the first method invocation recording component passed as the parameter to marshal the first method invocation into a data stream representative of the invocation into a message (claim 11, lines 48 – 60). As to a second invocation recording component, obviously multiple method invocation recording components could be created.

As to passing an object reference as a parameter, and including a data stream representation of the method invocations recording component to be marshaled into a method invocations message, see the double patenting rejection to claim 1 above.

6. Claim 16 is rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 13 of U.S. Patent No. 6,425,017 (herein after Dievendorff) in view of Wollrath.

As to claim 16, Dievendorff teaches a queued method invocations playing component operating to retrieve a method invocations message from a message queue associated with a first queued component, the method invocations playing component further operating in response to a message containing a data stream to a message containing a data stream representative of an invocation of the reference passing method, and to invoke the method on the first component with a reference for the re-created method invocation recording component passed as the parameter (claim 13, lines 7 – 18).

As to a reference passing method accepting a passed object reference as a parameter, a reference for a method invocation recording component of a second component passed as the parameter, and re-create the method invocation recording component, see the double patenting rejection to claim 1 above.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1 – 11 and 14 – 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,253,256 to Wollrath in view of U.S. Patent No. 5,848,234 to Chernick.

As to claim 16, Wollrath teaches a method invocation (remote method invocation; column 4, lines 25 – 38) system comprising:

a method invocations playing component (RMI 410 and code 411 for processing of objects, Fig. 4; column 6, lines 30 – 36) operating to receive a method invocations message (byte stream 408, Fig. 4) for a first component (receiving machine 402 contains its own RMI 410 and code 411 for processing of objects; column 6, lines 30 – 37), the first component having a reference passing method accepting a passed object reference as a parameter (a marshalled object is a container for an object that allows that object to be passed as a parameter in an RMI call...a container is an envelope that includes the data and either the code or a reference to the code for the object; column 6, lines 60 – 67), the method invocations playing component further operating in response to a message containing a data stream (byte stream 408) having a reference for a method invocation recording component of a second component passed as the parameter (contains a marshalled object) to unmarshal the data stream ("get" method is a method called by an application to execute a process of unmarshalling, which is reconstruction of an object from a marshalled object using a self-describing byte stream; column 7, lines 34 – 50), to re-create (create a new object 414) the method invocation recording component (if byte stream 408 contains a marshalled object, machine 402 may create a new object 414 using the object type identified in the marshalled object, the state information, and code for the object; column 6, lines 30 – 46 and 60 – 67), and to invoke the method on the first component with a reference for the re-created method invocation recording component passed as the parameter (the receiving machine holds the marshalled object for later use in response to a get method invoked by a process on the receiving machine...if the machine uses the object, it performs reconstruction of the object using its RMI and associated code, step 605, Fig. 6; column 8, lines 33 - 50). Wollrath does not teach queued method invocation.

However, Chernick teaches object procedure messaging for communication between objects (column 4, lines 11 – 18) and queued method invocation (server object 16 listens to message queue 22, using a conventional asynchronous RPC signal, to determine if an object handle designating server object 16 has been placed in message queue 22...when server object 16 determines that a message call to server object 16 has been placed in message queue 22, server object 16 causes that message call to be transferred to server object 16, Figs. 1 and 3; column 6, lines 27 – 64).

It would have been obvious to a person of ordinary skill in the art at the time of the invention to apply the teaching of queuing method invocations as taught by Chernick to the invention of Wollrath because the message queue is a very efficient and reliable transport mechanism that avoids some of the reduction in system performance inherent with conventional network data interchange techniques (column 2, lines 18 – 29 of Chernick).

As to claim 6, Wollrath teaches a client program issuing a first set of method invocations for the first component, marshaling data for the method invocations of the first set into a message (RMI 407 uses code 409 for converting object 405 into a marshalled object that is transmitted as a byte stream 408, Fig. 4; column 6, lines 1 – 10); and

marshaling an interface pointer reference to the second component in any of the method invocations issued by the client program for the first component, incorporating interface passing information (object's code or URL) in the data marshaled into the message (a marshalled object's constructor takes a serializable object (obj) as its single argument and holds the marshalled representation of the object in a byte stream, column 7, lines 33 – 49; the marshalled representation of the object preserves the semantics of objects that are at passed in RMI calls: each class in the stream is typically annotated with either the object's code or a URL to the code; column 7, lines 33 – 49). As to queued method invocations, see the rejection to claim 16.

As to claim 9, Wollrath teaches a client program issuing a set of method invocations for the component, marshaling data for the method invocations of the set into a message (RMI 407 uses code 409 for converting object 405 into a marshalled object that is transmitted as a byte stream 408, Fig. 4; column 6, lines 1 – 10);

marshaling a reference for calling methods on the persist-able object in any of the method invocations issued by the client program for the component, persisting the persist-able object (serializable object) into an object representative data stream and incorporating the object-representative data stream in the data marshaled into the message (a marshalled object's constructor takes a serializable object (obj) as its single argument and holds the marshalled representation of the object in a byte stream; column 7, lines 33 – 49); and

unmarshaling the data for the method invocations from the message ("get" method is a method called by an application to execute a process of unmarshalling, which is reconstruction of an object from a marshalled object using a self-describing byte stream; column 7, lines 34 – 50), re-creating the persist-able object from the object-representative data stream (if byte stream 408 contains a marshalled object, machine 402 may create a new object 414 using the object type identified in the marshalled object, the state information, and code for the object; column 6, lines 30 – 46 and 60 – 67), issuing the set of method invocations to the component, and passing a reference for calling methods on the re-created persist-able object in said any method invocation to the component (the receiving machine holds the marshalled object for later use in response to a get method invoked by a process on the receiving machine...if the machine uses the object, it performs reconstruction of the object using its RMI and associated code, step 605, Fig. 6; column 8, lines 33 - 50). As to queued method invocations, see the rejection to claim 16.

As to claim 14, Wollrath teaches a method invocation system comprising:

a component recorder constructor operating on request of a client program to obtain a component reference to create a method invocation recording component (objects such as objects 405 and 406) and return a reference for such method invocation recording component to the client program (transmitting machine 401 includes a memory 404 storing objects such as objects 405 and 406, and an RMI 407 for performing processing on the objects; column 5, line 62 - column 6, line 10);

a first method invocation recording component created by the component recorder constructor (objects such as objects 405 and 406; column 5, line 62 - column 6, line 10; object oriented objects would inherently be created by a constructor); and

a second method invocation recording component (RMI 407 uses code 409 for converting object 405 into a marshalled object that is transmitted as a byte stream 408, Fig. 4; column 6, lines 1 – 10) operating in response to an invocation of the reference passing method made on the second method invocation recording component having a reference for the first method invocation recording component passed as the parameter (a marshalled object's constructor takes a serializable object (obj) as its single argument and holds the marshalled representation of the object in a byte stream; column 7, lines

33 – 49) to marshal the first method invocation recording component into a data stream representative of the invocation into a message (the marshalled representation of the object preserves the semantics of objects that are at passed in RMI calls: each class in the stream is typically annotated with either the object's code or a URL to the code; column 7, lines 33 – 49). As to queued method invocations, see the rejection to claim 16.

As to claim 15, Wollrath teaches a method invocations playing component (RMI 410 and code 411 for processing of objects, Fig. 4; column 6, lines 30 – 36) operating to retrieve the message (byte stream 408, Fig. 4), the method invocations playing component further operating in response to the message to unmarshal the data stream ("get" method is a method called by an application to execute a process of unmarshalling, which is reconstruction of an object from a marshalled object using a self-describing byte stream; column 7, lines 34 – 50), to re-create the first method invocation recording component (if byte stream 408 contains a marshalled object, machine 402 may create a new object 414 using the object type identified in the marshalled object, the state information, and code for the object; column 6, lines 30 – 46 and 60 – 67), and to invoke the method on the second component with a reference for the re-created method invocation recording component passed as the parameter (the receiving machine holds the marshalled object for later use in response to a get method invoked by a process on the receiving machine...if the machine uses the object, it performs reconstruction of the object using its RMI and associated code, step 605, Fig. 6; column 8, lines 33 - 50). As to a queued method invocation, see the rejection to claim 16.

As to claim 1, Wollrath teaches a method invocation recording facility operative responsive to request of a client program to supply method invocations recorders for object instances of object classes (objects such as objects 405 and 406) designated as supporting method invocations (transmitting machine 401 includes a memory 404 storing objects such as objects 405 and 406, and an RMI 407 for performing processing on the objects; column 5, line 62 - column 6, line 10);

a first method invocations recorder supplied by the method invocation recording facility at request of the client program for the first object class (objects such as objects 405 and 406; column 5, line 62 - column 6, line 10); and

a second method invocations recorder supplied by the method invocation recording facility at request of the client program for the second object class (RMI 407 uses code 409 for converting object 405 into a marshalled object that is transmitted as a byte stream 408, Fig. 4; column 6, lines 1 – 10), the second method invocations recorder operating in response to a method invocation in which an object reference to the first method invocations recorder is passed (a marshalled object's constructor takes a serializable object (obj) as its single argument and holds the marshalled representation of the object in a byte stream; column 7, lines 33 – 49) to cause a data stream representation of the first method invocations recorder to be marshaled into a method invocations message for submission into a message queue associated with the second object class (the marshalled representation of the object preserves the semantics of objects that are at passed in RMI calls: each class in the stream is typically annotated with either the object's code or a URL to the code; column 7, lines 33 – 49);

a method invocation play-back facility operative responsive to a method invocations message to supply method invocation players (RMI 410 and code 411 for processing of objects, Fig. 4; column 6, lines 30 – 36) for object instances of object classes designated as supporting method invocations (receiving machine 402 contains its own RMI 410 and code 411 for processing of objects, Fig. 4; column 6, lines 30 – 37);

a first method invocation player (RMI 410 and code 411 for processing of objects, Fig. 4; column 6, lines 30 – 36) operating in response to the method invocations message to unmarshal the date stream representation ("get" method is a method called by an application to execute a process of unmarshalling, which is reconstruction of an object from a marshalled object using a self-describing byte stream; column 7, lines 34 – 50) and create therefrom a copy of the first method invocations recorder (if byte stream 408 contains a marshalled object, machine 402 may create a new object 414 using the object type identified in the marshalled object, the state information, and code for the object; column 6, lines 30 – 46 and 60 – 67), and to pass an object reference to the copy of the first method invocations recorder as a parameter of a method invocation to an object of the second object class (the receiving machine holds the marshalled object for later use in response to a get method invoked by a process on the receiving machine...if the machine uses the object, it performs reconstruction of the object using its RMI and associated code, step 605, Fig. 6; column 8, lines 33 - 50). As to queued method invocations, see the rejection to claim 16. Wollrath does not teach an object configuration store.

However, Chernick teaches an object configuration store (directory services) containing object properties information (information about resources) representing properties object classes (information such as objects and interfaces), the object properties information designating objects as supporting method invocation (global database for storing the services available from server objects that can be used and accessed), the object class having a method with a parameter for passing an object reference (directory services maintains information about resources, such as objects and interfaces, currently available in network...directory services includes a global database for storing the services available from server objects that can be used and accessed by client objects in network; column 9, lines 3 – 18).

It would have been obvious to a person of ordinary skill in the art at the time of the invention to apply the teaching of an object configuration store as taught by Chernick to the invention of Wollrath because this permits server objects to register and unregister themselves and provides look-up and listing functions for client objects in search of services (column 9, lines 13 – 16 of Chernick).

As to claim 2, Wollrath as modified teaches the first method invocations recorder is marshaled into the data stream representation via a marshal-by-value operation (marshalled object's constructor takes a serializable object (obj) as its single argument; column 7, lines 33 – 49 of Wollrath), such that the copy of the first method invocations recorder can be created on a separate computing machine ("get" method is a method called by an application to execute a process of unmarshalling, which is reconstruction of an object from a marshalled object using a self-describing byte stream; column 7, lines 33 – 49 of Wollrath).

As to claim 3, Wollrath as modified teaches a persistence interface associated with the first method invocations recorder operating when invoked to persistently write the data stream representation of the first method invocations recorder (a marshalled object is a container for an object that allows that object to be passed as a parameter in an RMI call; column 6, lines 60 – 67 of Wollrath), and the second method invocations recorder invoking the persistence interface to cause marshaling of the first method invocations recorder into the data stream representation (to transmit an object over network 400, RMI 407 uses code 409 for converting object 405 into a marshalled object that is transmitted as a byte stream 408 to machine 402; column 6, lines 1 – 10 of Wollrath).

As to 4, Wollrath as modified teaches the second method invocations recorder further operates to cause an identification (URL) of a message queue associated with the first object class to be marshaled into the data stream representation (to convert an object into a marshalled object, the object is placed inside a marshalled object container and when a URL is used to locate code for the object, the URL is added to the container; column 7, lines 8 – 15 of Wollrath).

As to clam 5, Wollrath as modified teaches execution of distributed objects (if the machine uses the object, it performs reconstruction of the object using its RMI and associated code, step 605, Fig. 6; column 8, lines 33 – 50 of Wollrath) across remote machines in a distributed computing system (transmitting objects in a distributed system having multiple machines; column 3, lines 33 – 42 of Wollrath).

As to claim 7, Wollrath as modified teaches responsive to the first queued component issuing a second set of method invocations, enqueueing the method invocations of the second set into the second message queue (message call placed on message queue 22 includes an object handle representing server object 16; column 6, lines 43 – 65 of Chernick).

As to claim 8, Wollrath as modified teaches passing the interface pointer reference in queued method invocations to multiple further queued components; and

responsive to the first queued component and the multiple queued components issuing sets of method invocations on the interface of the second queued component, enqueueing the method invocations of each such set into the second message queue (message call placed on message queue 22 includes an object handle representing server object 16; column 6, lines 43 – 65 of Chernick).

As to claim 10, Wollrath as modified teaches the persist-able object is a method invocations recorder of a second queued component (a marshalled object is a container for an object that allows that object to be passed as a parameter in an RMI call; column 6, lines 60 – 67 of Wollrath).

As to claim 11, Wollrath as modified teaches the references for calling methods are interface pointers referencing an interface exposed by the persist-able object (a marshalled object's constructor takes a serializable object (obj) as its single argument and holds the marshalled representation of the object in a byte stream; column 7, lines 33 – 65 of Wollrath).

9. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,567,861 to Kasichainula in view of Chernick.

As to claim 12, Kasichainula teaches a distributed computing system (distributed objects; column 3, lines 57 – 67; column 6, lines 24 – 50), a multiplicity of client machines (client system 401, Fig. 4B), a server machine (server system 402, Fig. 4B), a server-side component (server object Y 404, Fig. 4B), and a client-specific component (client object X 403, Fig. 4B);

a client program (object X 502) issuing a first method invocation to the server-side component (object Y 503) having passed therein a reference (object Z 504 is one of the parameters) for a client-specific component (Object X 502 makes a remote method call to object Y 503, and object Z 504 is one of the parameters, Fig. 5B; column 7, lines 55 - 67),

recording data representative of the first method invocation into a first method invocations message (object Y' 510 creates a proxy object Z" 513 for object Z 504 before passing the call to Y" 511), automatically and transparently marshaling a reference to the second message queue (reference to proxy object Z") with the data representative of the first method invocation into the first method invocations message (object Y' 510 creates a proxy object Z" 513 for object Z 504 before passing the call to Y" 511...this proxy contains the code necessary to allow a reference to itself to be passed over the network...object Y' passes the call to object Y" via Remote Method

Invocation or some other standard remote calling method 514, and a reference to Z" is provided in place of Z as the parameter in the call, Fig. 5B; column 8, lines 19 - 33);

submitting the first method invocations message to the first message queue (object Y' passes the call to object Y" via Remote Method Invocation or some other standard remote calling method 514; column 8, lines 19 – 33);

retrieving the first method invocations message from the first message queue at the server machine (upon receiving the remote call from Y' 510; column 8, lines 33 - 42);

unmarshaling the data representative of the first method invocation from the first method invocations message (Y" 511 creates a proxy Z' 512 for object Z 504 in machine 509, and creates 515 a reference table entry in which the key Z' returns a remote call reference to the proxy Z" 513 which was created by Y' 510...then when object Y" 511 translates the call into the semantics of object Y 503, Fig. 5B; column 33 – 42);

invoking per the first method invocation a method of the server-side queued component (invokes object Y), wherein said invoking comprises passing a reference for the client-specific queued component (reference to proxy object Z' 512 is passed as the parameter) to the server-side queued component (invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter...thus object Y will invoke object Z' 512 when object Y's invoked method invokes the object passed in as a parameter; column 8, lines 33 - 67); and on invoking by the server-side queued component a method of the client-specific queued component using the reference passed to the server side queued component (when object Z' is invoked 506 by object Y 503, object Z' uses the reference table entry which was created earlier 515 by object Y" 511 to determine where the call is to be directed; column 8, lines 33 – 67), automatically and transparently to the server-side queued component recording data representative of the server-side queued component's method invocations using the reference passed to the server side queued component into a second method invocations message (object Z' translates the call into the semantics of object Z" and invokes object Z" using Remote Method Invocation or some other standard remote calling method 518; column 8, lines 33 – 67), and submitting the second method invocations message to the second message queue, whereby the server-side queued component's method invocations are queued for the client-specific queued component (object Y finishes the method which was invoked from object X 502, it returns the result 523, if any, of said invocation to object Y" 511, which returns said result to object Y' 510, which returns 508 said result to object X 502, thus completing the object X's method invocation to object Y 503, which also updated object Z 504; column 8, lines 33 – 67). Kasichainula does not teach queued method invocation.

However, Chernick teaches object procedure messaging for communication between objects (column 4, lines 11 – 18) and queued method invocation (server object 16 listens to message queue 22, using a conventional asynchronous RPC signal, to determine if an object handle designating server object 16 has been placed in message queue 22...when server object 16 determines that a message call to server object 16 has been placed in message queue 22, server object 16 causes that message call to be transferred to server object 16, Figs. 1 and 3; column 6, lines 27 – 64).

It would have been obvious to a person of ordinary skill in the art at the time of the invention to apply the teaching of queuing method invocations as taught by Chernick to the invention of Kasichainula because the message queue is a very efficient and reliable transport mechanism that avoids some of the reduction in system performance inherent with conventional network data interchange techniques (column 2, lines 18 – 29 of Chernick).

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

U.S. Patent No. 6,125,400 to Cohen teaches passing objects as a parameter in a remote method invocation.

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Li B. Zhen whose telephone number is (703) 305-3406. The examiner can normally be reached on Mon - Fri, 8am - 4:30pm.

The fax phone numbers for the organization where this application or proceeding is assigned are (703) 746-7239 for regular communications and (703) 746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Li B. Zhen
Examiner
Art Unit 2126

lbz
June 20, 2003

[illegible]